Appendix C

USING REFPROP WITH OTHER PROGRAMS
FORTRAN SOURCE CODE

The subroutines are contained in the following files which are placed in the FORTRAN directory by the installation script. They should be compiled and linked with your own main program:

core_ANC.for	core_BWR.for	core_CPP.for
core_DE.for	core_ECS.for	core_FEQ.for
cor_MLT.for	core_PH0.for	core_STN.for
flash2.for	flsh_sub.for	idealgas.for
mix_HMX.for	prop_sub.for	realgas.for
sat_sub.for	setup.for	setup2.for
trnsp.for	trns_ECS.for	trns_TCX.for
trns_VIS.for	utility.for	

All the above files are required.

Several additional files are provided. The file “example.for” is a main program providing simple examples of calling the property routines. The file “pass_ftn.for” contains the interface routines used by the graphical interface and dynamic link libraries; the file ftn_pas.for is similar and is included for compatibility with Version 6. They would not be needed by ordinary FORTRAN applications. The file “X_sub.for” contains alternative versions of the most commonly called REFPROP routines. These routines are the same as the standard ones except that they have only integer and real arguments (no character strings) and so are intented for cross-language applications that cannot handle strings conveniently (e.g., LabView).

Any fluids which are used must have a corresponding “.fld” file, *e.g.* R134a.fld or propane.fld. The file “HMX.bnc” (containing mixture parameters) must also be present. It is suggested (but not required) that these be put into a subdirectory called “fluids” under the directory containing the application program.

The subroutine SETUP must be called to initialize the pure fluid or mixture components. The call to SETUP will allow the choice of one of several standard reference states for entropy and enthalpy and will automatically load the "NIST-recommended" models for the components as well as mixing rules. The subroutine SETMOD allows the specification of other models. The subroutine SETKTV allows the specification of mixing rules and parameters. To define another reference state, or to apply one of the standard states to a mixture of a specified composition, the subroutine SETREF may be used. These subroutines should be called only if the fluids and/or models (or reference state) are changed.

The sequence is:

call SETMOD	(optional)
call SETUP	(REQUIRED)
call SETKTV	(optional)
call SETREF	(optional)

C.1 Units

All inputs and outputs to the subroutines are in the following units:

temperature	K
pressure, fugacity	kPa
density	mol/L
composition	mole fraction
quality	mole basis
(except mass basis allowed when quality is input)	
enthalpy, internal energy	J/mol
Gibbs, Helmholtz free energy	J/mol
entropy, heat capacity	J/(mol·K)
speed of sound	m/s
Joule-Thompson coefficient	K/kPa
d(p)/d(rho)	kPa·L/mol
$d^2(p)/d(rho)^2$	kPa·(L/mol)^2
viscosity	μPa·s (10^{-6} Pa·s)
thermal conductivity	W/(m·K)
dipole moment	debye
surface tension	N/m

C.2 Naming conventions

The variable type of subroutine arguments can generally be inferred from the first letter of the variable name:

a-g and o-z:	double precision
h:	double precision (*i.e.* enthalpy) or character
i-k and m,n:	integer
l:	logical (within subroutines only, no logicals are used in arguments) also used as integer for exponent of FEQ model

The property subroutines are compatible with FORTRAN 90. The routines are largely compatible with ANSI standard FORTRAN 77; the only extension widely used is the “do -- enddo” construct, which is supported by most FORTRAN 77 compilers. We have tested the source code on the following compilers:

Absoft Pro FORTRAN 5.0
Digital Visual FORTRAN 5.0
Fortner Research FORTRAN (Macintosh)
Lahey FORTRAN77
Lahey FORTRAN90
Lahey/Fujitsu LF95
Microsoft FORTRAN 5.1
Microsoft PowerStation FORTRAN 4.0
Salford Software FTN77/x86

We are striving to make these subroutines as standard and portable as possible, but every compiler has its own sensitive points. Please report any compiler or linker errors or warnings.

Potential pitfalls:

The fluid data files are read using logical unit 12. Use of this unit in your application program may crash the application program and/or the subroutines and should be avoided.

Following is a description of the high-level subroutines that would be used in stand-alone applications. These subroutines will give access to all features in a model-independent fashion. (There are corresponding low-level subroutines for some of these which call specific models. Please do not incorporate the low-level subroutines into your applications—if you do, you may find that future versions may not work the same.)

C.3 Initialization Subroutines

```
      subroutine SETUP (nc,hfiles,hfmix,hrf,ierr,herr)
c
c  define models and initialize arrays
c
c  A call to this routine is required.
c
c  inputs:
c       nc--number of components (1 for pure fluid) [integer]
c   hfiles--array of file names specifying fluid/mixture
components
c           [character*255 variable] for each of the nc
components;
c           e.g., :fluids:R134a.fld (Mac) or fluids\R134a.fld
(DOS) or
c           [full_path]/fluids/R134a.fld (UNIX)
c    hfmix--mixture coefficients [character*255]
c           file name containing coefficients for mixture
model,
c           e.g.,  :fluids:HMX.bnc
c      hrf--reference state for thermodynamic calculations
[character*3]
c           'DEF':  default reference state as specified in
fluid file
c                   is applied to each pure component
c           'NBP':  h,s = 0 at pure component normal boiling
point(s)
c           'ASH':  h,s = 0 for sat liquid at -40 C (ASHRAE
convention)
c           'IIR':  h = 200, s = 1.0 for sat liq at 0 C (IIR
convention)
c           other choices are possible, but these require a
separate
c           call to SETREF
c  outputs:
c     ierr--error flag:  0 = successful
c                      101 = error in opening file
c                      102 = error in file or premature end of
file
c                     -103 = unknown model encountered in file
c                      104 = error in setup of model
c                      105 = specified model not found
c                      111 = error in opening mixture file
c                      112 = mixture file of wrong type
c     herr--error string (character*255 variable if ierr<>0)
c     [fluid parameters, etc. returned via various common
blocks]
```

```
      subroutine SETMOD (nc,htype,hmix,hcomp,ierr,herr)
c
c  set model(s) other than the NIST-recommended ('NBS') ones
c
c  This subroutine must be called before SETUP; it need not be
called
c  at all if the default (NIST-recommended) models are
desired.
c
c  inputs:
c       nc--number of components (1 for pure fluid) [integer]
c    htype--flag indicating which models are to be set
[character*3]
c           'EOS':  equation of state for thermodynamic
properties
c           'ETA':  viscosity
c           'TCX':  thermal conductivity
c           'STN':  surface tension
c           'NBS':  reset all of the above model types and all
c                   subsidiary component models to 'NBS';
c                   values of hmix and hcomp are ignored
c     hmix--mixture model to use for the property specified in
the
c           argument htype [character*3];
c           this can a pure fluid model if number of
components = 1
c           'NBS':  use NIST recommendation for specified
fluid/mixture
c                   (valid input for all properties, pure or
mix)
c
c           some allowable choices for thermodynamic
properties:
c           pure fluids:
c           'FEQ':  Helmholtz free energy model
c           'BWR':  pure fluid modified Bennedict-Webb-Rubin
(MBWR)
c           'ECS':  pure fluid thermo extended corresponding
states
c           mixture:
c           'HMX':  mixture Helmholtz model for thermodynamic
properties
c
c           some allowable choices for viscosity:
c           pure fluids:
c           'ECS':  extended corresponding states (all fluids)
c           'VS1':  the 'composite' model for R134a, R152a,
NH3, etc.
c           'VS2':  Younglove-Ely model for hydrocarbons
c           mixture:
c           'ECS':  extended corresponding states
```

```
c
c           some allowable choices for thermal conductivity:
c           pure fluids:
c           'ECS':  extended corresponding states (all fluids)
c           'TC1':  the 'composite' model for R134a, R152a,
etc.
c           'TC2':  Younglove-Ely model for hydrocarbons
c           mixture:
c           'ECS':  extended corresponding states
c
c           some allowable choices for surface tension:
c           pure fluids:
c           'ST1':  surface tension as f(tau); tau = 1 - T/Tc
c           mixture:
c           'STX':  surface tension mixture model
c
c    hcomp--component model(s) to use for property specified
in htype
c           [array (1..nc) of character*3]
c           'NBS':  NIST recommendation for specified
fluid/mixture
c                   (valid input for all properties, pure or
mix)
c                   other choices are identical to the pure
fluid
c                   choices listed above
c  outputs:
c     ierr--error flag:  0 = successful
c                        no errors are returned from this
routine,
c                        ierr and herr included to maintain
parallel
c                        structure and for possible future use
c     herr--error string (character*255 variable if ierr<>0)
c     [fluid parameters, etc. returned via various common
blocks]
```

```
      subroutine SETKTV
(icomp,jcomp,hmodij,fij,hfmix,ierr,herr)
c
c  set mixture model and/or parameters
c
c  This subroutine must be called after SETUP, but before any
call to
c  SETREF; it need not be called at all if the default mixture
c  parameters (those read in by SETUP) are to be used.
c
c  inputs:
c    icomp--component i
c    jcomp--component j
c   hmodij--mixing rule for the binary pair i,j [character*3]
c           e.g. 'LJ1' (Lemmon-Jacobsen model) or
c                'LIN' (linear mixing rules)
c                'LM1' (modified Lemmon-Jacobson model)
c           'RST' indicates reset all pairs to values from
c                 original call to SETUP (i.e. those read from
file)
c                 [all other inputs are ignored]
c      fij--binary mixture parameters [array of dimension
nmxpar;
c           currently nmxpar is set to 6]
c           the parameters will vary depending on hmodij;
c           for example, for the Lemmon-Jacobsen model (LJ1):
c             fij(1) = zeta
c             fij(2) = xi
c             fij(3) = Fpq
c             fij(4) = beta
c             fij(5) = gamma
c             fij(6) = 'not used'
c    hfmix--file name [character*80] containing generalized
parameters
c           for the binary mixture model; this will usually be
the same
c           as the corresponding input to SETUP
(e.g.,':fluids:HMX.bnc')
c  outputs:
c     ierr--error flag:  0 = successful
c                      111 = error in opening mixture file
c                      112 = mixture file of wrong type
c                     -113 = illegal i,j specification
c                            (i = j or i > nc or j > nc)
c     herr--error string (character*255 variable if ierr<>0)
c     [mixture parameters returned via various common blocks]
```

```
      subroutine SETREF (hrf,ixflag,x0,h0,s0,t0,p0,ierr,herr)
c
c  set reference state enthalpy and entropy
c
c  This subroutine must be called after SETUP; it need not be
called at
c  all if the reference state specified in the call to SETUP
is to be
c  used.
c
c  inputs:
c      hrf--reference state for thermodynamic calculations
[character*3]
c           'NBP':  h,s = 0 at normal boiling point(s)
c           'ASH':  h,s = 0 for sat liquid at -40 C (ASHRAE
convention)
c           'IIR':  h = 200, s = 1.0 for sat liq at 0 C (IIR
convention)
c           'DEF':  default reference state as specified in
fluid file
c                   is applied to each component (ixflag = 1
is used)
c           'OTH':  other, as specified by h0, s0, t0, p0
c   ixflag--composition flag:  1 = ref state applied to pure
components
c                              2 = ref state applied to
mixture x0
c  following input has meaning only if ixflag = 2
c       x0--composition for which h0, s0 apply; array(1:nc)
[mol frac]
c           this is useful for mixtures of a predefined
composition,
c           e.g. refrigerant blends such as R410A
c  following inputs have meaning only if hrf = 'OTH'
c       h0--reference state enthalpy at t0,p0 {x0} [J/mol]
c       s0--reference state entropy at t0,p0 {x0} [J/mol-K]
c       t0--reference state temperature [K]
c           t0 = -1 indicates saturated liquid at normal
boiling point
c                   (bubble point for a mixture)
c       p0--reference state pressure [kPa]
c           p0 = -1 indicates saturated liquid at t0 {and x0}
c           p0 = -2 indicates saturated vapor at t0 {and x0}
c  outputs:
c     ierr--error flag:  0 = successful
c                       22 = Tmin > Tref for IIR reference
state
c                       23 = Tcrit < Tref for IIR reference
state
c                       24 = Tmin > Tref for ASHRAE reference
state
```

```
c                        25 = Tcrit < Tref for ASHRAE reference
state
c                        26 = Tmin > Tnbp for NBP reference
state
c                        27 = Tref, Pref for OTH ref state
outside limits
c                       -28 = can't apply 'DEF' to mixture;
c                             will apply to pure components
c                       -29 = unknown reference state
specified;
c                             will use 'DEF'
c     herr--error string (character*255 variable if ierr<>0)
c     [fluid parameters, etc. returned via various common
blocks]
```

```
      subroutine GETKTV
(icomp,jcomp,hmodij,fij,hfmix,hfij,hbinp,hmxrul)
c
c  retrieve mixture model and parameter info for a specified
binary
c
c  This subroutine should not be called until after a call to
SETUP.
c
c  inputs:
c    icomp--component i
c    jcomp--component j
c  outputs:
c   hmodij--mixing rule for the binary pair i,j (e.g. LJ1 or
LIN)
c           [character*3]
c      fij--binary mixture parameters [array of dimension
nmxpar;
c           currently nmxpar is set to 6]; the parameters will
vary
c           depending on hmodij;
c    hfmix--file name [character*255] containing parameters
for the
c           binary mixture model
c     hfij--description of the binary mixture parameters
[character*8
c           array of dimension nmxpar]
c           for example, for the Lemmon-Jacobsen model (LJ1):
c             fij(1) = zeta
c             fij(2) = xi
c             fij(3) = Fpq
c             fij(4) = beta
c             fij(5) = gamma
c             fij(6) = 'not used'
c    hbinp--documentation for the binary parameters
[character*255]
c           terminated with ASCII null character
c   hmxrul--description of the mixing rule [character*255]
```

C.4 Saturation-State Subroutines

```
      subroutine SATT
(t,x,kph,p,rhol,rhov,xliq,xvap,ierr,herr)
c
c  iterate for saturated liquid and vapor states given
temperature
c  and the composition of one phase
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac] (phase specified
by kph)
c      kph--phase flag: 1 = input x is liquid composition
(bubble point)
c                       2 = input x is vapor composition (dew
point)
c                       3 = input x is liquid comp (freezing
point)
c                       4 = input x is vapor comp (sublimation
point)
c  outputs:
c        p--pressure [kPa]
c     rhol--molar density [mol/L] of saturated liquid
c     rhov--molar density [mol/L] of saturated vapor
c     xliq--liquid phase composition [array of mol frac]
c     xvap--vapor phase composition [array of mol frac]
c     ierr--error flag:   0 = successful
c                         1 = T < Tmin
c                         8 = x out of range
c                         9 = T and x out of range
c                       120 = CRITP did not converge
c                       121 = T > Tcrit
c                       122 = TPRHO-liquid did not converge
(pure fluid)
c                       123 = TPRHO-vapor did not converge
(pure fluid)
c                       124 = pure fluid iteration did not
converge
c           following 3 error codes are advisory--iteration
will either
c           converge on later guess or error out (ierr = 128)
c                      -125 = TPRHO did not converge for
parent ph (mix)
c                      -126 = TPRHO did not converge for
incipient (mix)
c                      -127 = composition iteration did not
converge
c                       128 = mixture iteration did not
converge
c     herr--error string if ierr<>0 (character*255)
```

```
      subroutine SATP
(p,x,kph,t,rhol,rhov,xliq,xvap,ierr,herr)
c
c  iterate for saturated liquid and vapor states given
temperature
c  and the composition of one phase
c
c  inputs:
c        p--pressure [kPa]
c        x--composition [array of mol frac] (phase specified
by kph)
c      kph--phase flag:  1 = input x is liquid composition
c                        2 = input x is vapor composition
c
c  outputs:
c        t--temperature [K]
c     rhol--molar density [mol/L] of saturated liquid
c     rhov--molar density [mol/L] of saturated vapor
c     xliq--liquid phase composition [array of mol frac]
c     xvap--vapor phase composition [array of mol frac]
c     ierr--error flag:  0 = successful
c                        4 = P < 0
c                        8 = x out of range
c                       12 = P and x out of range
c                      140 = CRITP did not converge
c                      141 = P > Pcrit
c                      142 = TPRHO-liquid did not converge
(pure fluid)
c                      143 = TPRHO-vapor did not converge
(pure fluid)
c                      144 = pure fluid iteration did not
converge
c           following 3 error codes are advisory--iteration
will either
c           converge on later guess or error out (ierr = 148)
c                     -144 = Raoult's law (mixture intial
guess) did
c                            not converge
c                     -145 = TPRHO did not converge for parent
ph (mix)
c                     -146 = TPRHO did not converge for
incipient (mix)
c                     -147 = composition iteration did not
converge
c                      148 = mixture iteration did not
converge
c     herr--error string if ierr<>0 (character*255)
```

```
      subroutine SATD
(rho,x,kph,kr,t,p,rhol,rhov,xliq,xvap,ierr,herr)
c
c  iterate for temperature and pressure given a density along
the
c  saturation boundary and the composition
c
c  inputs:
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c      kph--flag specifying desired root for multi-valued
inputs
c           has meaning only for water at temps close to its
triple point
c          -1 = return middle root (between 0 and 4 C)
c           1 = return highest temperature root (above 4 C)
c           3 = return lowest temperature root (along freezing
line)
c  outputs:
c        t--temperature [K]
c        p--pressure [kPa]
c     rhol--molar density [mol/L] of saturated liquid
c     rhov--molar density [mol/L] of saturated vapor
c     xliq--liquid phase composition [array of mol frac]
c     xvap--vapor phase composition [array of mol frac]
c       kr--phase flag: 1 = input state is liquid
c                       2 = input state is vapor in
equilibrium with liq
c                       3 = input state is liq in equilibrium
with solid
c                       4 = input state is vap in equilibrium
with solid
c     ierr--error flag:   0 = successful
c                         2 = D > Dmax
c                         8 = x out of range
c                        10 = D and x out of range
c                       160 = CRITP did not converge
c                       161 = SATD did not converge
c     herr--error string (character*255 variable if ierr<>0)
c
c  N.B. kr = 3,4 presently working only for pure components
c
c  either (rhol,xliq) or (rhov,xvap) will correspond to the
input state
c  with the other pair corresponding to the other phase in
equilibrium
c  with the input state
```

```
      subroutine SATH
(h,x,kph,nroot,k1,t1,p1,d1,k2,t2,p2,d2,ierr,herr)
c
c  iterate for temperature, pressure, and density given an
enthalpy along
c  the saturation boundary and the composition
c
c  inputs:
c        h--molar enthalpy [J/mol]
c        x--composition [array of mol frac]
c      kph--flag specifying desired root
c           0 = return all roots along the liquid-vapor line
c           1 = return only liquid VLE root
c           2 = return only vapor VLE roots
c           3 = return liquid SLE root (melting line)
c           4 = return vapor SVE root (sublimation line)
c  outputs:
c    nroot--number of roots.  Set to one for kph=1,3,4 if
ierr=0
c       k1--phase of first root (1-liquid, 2-vapor, 3-melt, 4-
subl)
c       t1--temperature of first root [K]
c       p1--pressure of first root [kPa]
c       dl--molar density of first root [mol/L]
c       k2--phase of second root (1-liquid, 2-vapor, 3-melt,
4-subl)
c       t2--temperature of second root [K]
c       p2--pressure of second root [kPa]
c       d2--molar density of second root [mol/L]
c     ierr--error flag:   0 = successful
c                         2 = h < hmin
c                         4 = h > hmax
c                         8 = h > htrp (for subl input)
c                       160 = CRITP did not converge
c                       161 = SATH did not converge for one
root
c                       162 = SATH did not converge for both
roots
c     herr--error string (character*255 variable if ierr<>0)
c
c  The second root is always set as the root in the vapor at
temperatures
c  below the maximum enthalpy on the vapor saturation line.
If kph = 2,
c  and only one root is found in the vapor (this occurs when
h<hcrit)
c  the state point will be placed in k2,t2,p2,d2.  If kph = 0
and this
c  situation occured, the first root (k1,t1,p1,d1) would be in
the liquid
c  (k1 =1, k2 = 2).
```

```
c
c  N.B. kph = 3,4 presently working only for pure components
```

```
      subroutine SATS (s,x,kph,nroot,k1,t1,p1,d1,k2,t2,p2,d2,
     &                 k3,t3,p3,d3,ierr,herr)
c
c  iterate for temperature, pressure, and density given an
entropy along
c  the saturation boundary and the composition
c
c  inputs:
c        s--molar entropy [J/mol-K]
c        x--composition [array of mol frac]
c      kph--flag specifying desired root
c           0 = return all roots along the liquid-vapor line
c           1 = return only liquid VLE root
c           2 = return only vapor VLE roots
c           3 = return liquid SLE root (melting line)
c           4 = return vapor SVE root (sublimation line)
c  outputs:
c    nroot--number of roots.  Set to one for kph=1,3,4 if
ierr=0
c       k1--phase of first root (1-liquid, 2-vapor, 3-melt, 4-
subl)
c       t1--temperature of first root [K]
c       p1--pressure of first root [kPa]
c       dl--molar density of first root [mol/L]
c       k2--phase of second root (1-liquid, 2-vapor, 3-melt,
4-subl)
c       t2--temperature of second root [K]
c       p2--pressure of second root [kPa]
c       d2--molar density of second root [mol/L]
c       k3--phase of thrid root (1-liquid, 2-vapor, 3-melt, 4-
subl)
c       t3--temperature of third root [K]
c       p3--pressure of third root [kPa]
c       d3--molar density of third root [mol/L]
c     ierr--error flag:   0 = successful
c                         2 = s < smin
c                         4 = s > smax
c                         8 = s > strp (for subl input)
c                       160 = CRITP did not converge
c                       161 = SATS did not converge for one
root
c                       162 = SATS did not converge for two
roots
c                       163 = SATS did not converge for all
roots
c     herr--error string (character*255 variable if ierr<>0)
c
c  The second root is always set as the root in the vapor at
temperatures
c  below the maximum entropy on the vapor saturation line.  If
kph = 2,
```

```
c  and only one root is found in the vapor (this occurs when
s<scrit)
c  the state point will be placed in k2,t2,p2,d2.  If kph = 0
and this
c  situation occured, the first root (k1,t1,p1,d1) would be in
the liquid
c  (k1=1, k2=2).
c
c  The third root is the root with the lowest temperature.
c  For fluids with multiple roots:  When only one root is
found in the
c  vapor phase (this happens only at very low temperatures
past the
c  region where three roots are located), the value of the
root is still
c  placed in k3,t3,p3,d3.  For fluids that never have more
than one root
c  (when there is no maximum entropy along the saturated vapor
line), the
c  value of the root is always placed in k1,t1,p1,d1.
c
c  N.B. kph = 3,4 presently working only for pure components



      subroutine CSATK (icomp,t,kph,p,rho,csat,ierr,herr)
c
c  compute the heat capacity along the saturation line as a
function of
c  temperature for a given component
c
c  csat can be calculated two different ways:
c     Csat = Cp - T(DvDT)(DPDTsat)
c     Csat = Cp - beta/rho*hvap/(vliq - vvap)
c     where beta is the volulme expansivity
c
c  inputs:
c    icomp--component number in mixture (1..nc); 1 for pure
fluid
c        t--temperature [K]
c      kph--phase flag: 1 = liquid calculation
c                       2 = vapor calculation
c  outputs:
c        p--saturation pressure [kPa]
c      rho--saturation molar density [mol/L]
c     csat--saturation heat capacity [J/mol-K]
```

Two similar routines are provided for calculating surface tension. SURTEN is more efficient if the liquid and vapor density and composition are known (e.g. from a previous call to SATT). If these are not known, then SURFT may be used.

```
      subroutine SURFT (t,rhol,xl,sigma,ierr,herr)
c
c  compute surface tension
c
c  inputs:
c        t--temperature [K]
c       xl--composition of liquid phase [array of mol frac]
c  outputs:
c     rhol--molar density of liquid phase [mol/L]
c    sigma--surface tension [N/m]
c     ierr--error flag:   0 = successful
c                         1 = T < Tmin
c                         8 = x out of range
c                         9 = T and x out of range
c                       120 = CRITP did not converge
c                       121 = T > Tcrit
c                       122 = TPRHO-liquid did not converge in
SATT
c                       123 = TPRHO-vapor did not converge in
SATT
c                       124 = SATT pure fluid iteration did
not converge
c                       128 = SATT mixture iteration did not
converge
c     herr--error string if ierr<>0 (character*255)


      subroutine SURTEN (t,rhol,rhov,xl,xv,sigma,ierr,herr)
c
c  compute surface tension
c
c  inputs:
c        t--temperature [K]
c     rhol--molar density of liquid phase [mol/L]
c     rhov--molar density of vapor phase [mol/L]
c           if either rhol or rhov < 0 call SATT to find
densities
c       xl--composition of liquid phase [array of mol frac]
c       xv--composition of liquid phase [array of mol frac]
c           (xv is optional input if rhol < 0 or rhov < 0)
c  outputs:
c    sigma--surface tension [N/m]
c     ierr--error flag:   0 = successful
c                         [all error codes identical to those
for SURFT]
c     herr--error string if ierr<>0 (character*255)
```

C.5 Flash Subroutines

So-called “flash” calculations involve a determination of the thermodynamic state given two independent variables plus composition. In addition to the inputs of temperature and pressure which is the most common flash calculation, REFPROP provides routines for virtually all combinations of temperature, pressure, or density as the first variable and density, internal energy, enthalpy, entropy, or quality as the second variable. The combination of enthalpy and entropy is also supported.

Flash subroutines are provided for cases where the state is known to be single phase (liquid or vapor), known to be two-phase (liquid plus vapor), and also for the general case where the phase is not known. Because of the many combinations and their parallel structure, these routines are described in groups. The first two letters of the subroutine name indicate the independent variables, where

```
T = temperature [K]
P = pressure [kPa]
D = density [mol/L]
E = internal energy [J/mol]
H = enthalpy [J/mol]
S = entropy [J/mol-K]
Q = vapor quality [moles vapor/total moles]
    or [kg vapor/total kg] depending on the
    value of the input flag kq
```

C.5.1 General flash subroutines

For cases where the phase is not known, the following routines are available.

```
      subroutine TPFLSH
(t,p,z,D,Dl,Dv,x,y,q,e,h,s,cv,cp,w,ierr,herr)
      subroutine TDFLSH
(t,D,z,p,Dl,Dv,x,y,q,e,h,s,cv,cp,w,ierr,herr)
      subroutine TEFLSH
(t,e,z,kr,p,D,Dl,Dv,x,y,q,h,s,cv,cp,w,ierr,herr)
      subroutine THFLSH
(t,h,z,kr,p,D,Dl,Dv,x,y,q,e,s,cv,cp,w,ierr,herr)
      subroutine TSFLSH
(t,s,z,kr,p,D,Dl,Dv,x,y,q,e,h,cv,cp,w,ierr,herr)
      subroutine PDFLSH
(p,D,z,t,Dl,Dv,x,y,q,e,h,s,cv,cp,w,ierr,herr)
      subroutine PEFLSH
(p,e,z,t,D,Dl,Dv,x,y,q,h,s,cv,cp,w,ierr,herr)
      subroutine PHFLSH
(p,h,z,t,D,Dl,Dv,x,y,q,e,s,cv,cp,w,ierr,herr)
      subroutine PSFLSH
(p,s,z,t,D,Dl,Dv,x,y,q,e,h,cv,cp,w,ierr,herr)
      subroutine DHFLSH
(D,h,z,t,p,Dl,Dv,x,y,q,e,s,cv,cp,w,ierr,herr)
      subroutine DSFLSH
(D,s,z,t,p,Dl,Dv,x,y,q,e,h,cv,cp,w,ierr,herr)
      subroutine DEFLSH
(D,e,z,t,p,Dl,Dv,x,y,q,h,s,cv,cp,w,ierr,herr)
      subroutine HSFLSH
(h,s,z,t,p,D,Dl,Dv,x,y,q,e,cv,cp,w,ierr,herr)
      subroutine TQFLSH
(t,q,z,kq,p,D,Dl,Dv,x,y,e,h,s,cv,cp,w,ierr,herr)
      subroutine PQFLSH
(p,q,z,kq,t,D,Dl,Dv,x,y,e,h,s,cv,cp,w,ierr,herr)
c
c  flash calculation given two independent variables and bulk
composition
c
c  These routines accept both single-phase and two-phase
states as the
c  input; if the phase is known, the specialized routines are
faster
c
c  inputs--two of the following as indicated by the first two
letters of
c          the subroutine name:
c        t--temperature [K]
c        p--pressure [kPa]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
```

```
c        s--entropy [[J/mol-K]
c        q--vapor quality [basis specified by kq]
c           q = 0 indicates saturated liquid
c           q = 1 indicates saturated vapor
c           q < 0 or q > 1 are not allowed and will result in
warning

c  additional input--required for all routines
c        z--overall (bulk) composition [array of mol frac]

c  additional input--only for TQFLSH and PQFLSH
c       kq--flag specifying units for input quality
c           kq = 1 quality on MOLAR basis [moles vapor/total
moles]
c           kq = 2 quality on MASS basis [mass vapor/total
mass]
c
c  outputs--one, two, or all of the following, depending on
the inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c        D--overall (bulk) molar density [mol/L]
c
c  additional outputs--common to all routines
c       Dl--molar density [mol/L] of the liquid phase
c       Dv--molar density [mol/L] of the vapor phase
c           if only one phase is present, Dl = Dv = D
c        x--composition of liquid phase [array of mol frac]
c        y--composition of vapor phase [array of mol frac]
c           if only one phase is present, x = y = z
c
c  additional output--common to all routines except TQFLSH and
PQFLSH
c        q--vapor quality on a MOLAR basis [moles vapor/total
moles]
c           q < 0 indicates subcooled (compressed) liquid
c           q = 0 indicates saturated liquid
c           q = 1 indicates saturated vapor
c           q > 1 indicates superheated vapor
c           q = 998 superheated vapor, but quality not defined
(t > Tc)
c           q = 999 indicates supercritical state (t > Tc) and
(p > Pc)
c
c  additional outputs--common to all routines, except that
input
c                      quantities are not repeated
c        e--overall (bulk) internal energy [J/mol]
c        h--overall (bulk) enthalpy [J/mol]
c        s--overall (bulk) entropy [J/mol-K]
c       Cv--isochoric (constant V) heat capacity [J/mol-K]
```

```
c       Cp--isobaric (constant p) heat capacity [J/mol-K]
c        w--speed of sound [m/s]
c           Cp, w are not defined for 2-phase states
c           in such cases, a flag = -9.99998d6 is returned
c     ierr--error flag:  0 = successful
c                        1 = Tin < Tmin
c                        4 = Pin < 0
c                        5 = T and P out of range
c                        8 = x out of range (component and/or
sum < 0
c                           or > 1)
c                        9 = x and T out of range
c                       12 = x out of range and P < 0
c                       13 = x and T and P out of range
c     herr--error string (character*255 variable if ierr<>0)
```

C.5.2 Single-phase flash subroutines

These routines accept only single-phase states as inputs. They will be faster than the corresponding general routines, but will fail if called with an incorrect phase specification. The phase-specific subroutines also do not check limits, so may fail if called outside the range of the equation of state. The following single-phase routines are available.

```
      subroutine TEFL1 (t,e,z,Dmin,Dmax,D,ierr,herr)
      subroutine THFL1 (t,h,z,Dmin,Dmax,D,ierr,herr)
      subroutine TSFL1 (t,s,z,Dmin,Dmax,D,ierr,herr)
      subroutine HSFL1 (h,s,z,Dmin,Dmax,t,D,ierr,herr)
c
c  inputs--two of the following as indicated by the first two
letters of c          the subroutine name:
c        t--temperature [K]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]

c  additional inputs--required for all routines
c        z--overall (bulk) composition [array of mol frac]
c     Dmin--lower bound on density [mol/L]
c     Dmax--upper bound on density [mol/L]
c
c  outputs:
c        t--temperature [K] (present only for HSFL1)
c        D--molar density [mol/L]
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)
```

Single-phase flash routines (continued)

```
      subroutine PEFL1 (p,e,z,kph,t,D,ierr,herr)
      subroutine PHFL1 (p,h,z,kph,t,D,ierr,herr)
      subroutine PSFL1 (p,s,z,kph,t,D,ierr,herr)
c
c  inputs--two of the following as indicated by the first two
letters of
c          the subroutine name:
c        p--pressure [kPa]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]

c  additional inputs--required for all routines
c        z--overall (bulk) composition [array of mol frac]
c      kph--phase flag:  1 = liquid
c                        2 = vapor
c
c  outputs:
c        t--temperature [K]
c        D--molar density [mol/L]
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)


      subroutine DEFL1 (D,e,z,t,ierr,herr)
      subroutine DHFL1 (D,h,z,t,ierr,herr)
      subroutine DSFL1 (D,s,z,t,ierr,herr)
      subroutine PDFL1 (p,D,z,t,ierr,herr)
c
c  inputs--two of the following as indicated by the first two
letters of
c          the subroutine name:
c        p--pressure [kPa]
c        D--density [mol/L]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]

c  additional inputs--required for all routines
c        z--overall (bulk) composition [array of mol frac]
c
c  outputs:
c        t--temperature [K]
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)
```

The following single-phase temperature-pressure flash is called many times by other routines and has been optimized for speed and requires a specific calling sequence.

```
      subroutine TPRHO (t,p,x,kph,kguess,rho,ierr,herr)
c
c  iterate for density as a function of temperature, pressure,
and
c  composition for a specified phase
c
c  inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c        x--composition [array of mol frac]
c      kph--phase flag:  1 = liquid
c                        2 = vapor
c                N.B.:  0 = stable phase--NOT ALLOWED (use
TPFLSH)
c   kguess--input flag:  1 = first guess for rho provided
c                        0 = no first guess provided
c      rho--first guess for molar density [mol/L], only if
kguess = 1
c
c  outputs:
c      rho--molar density [mol/L]
c     ierr--error flag:  0 = successful
c                      200 = CRITP did not converge
c                      201 = illegal input (kph <= 0)
c                         202 = liquid-phase iteration did not
converge
c                          203 = vapor-phase iteration did not
converge
c     herr--error string (character*255 variable if ierr<>0)
```

C.5.3 Two-phase flash subroutines

These routines accept only two-phase (liquid + vapor) states as inputs. They will be faster than the corresponding general routines, but will fail if called with an incorrect phase specification. The phase-specific subroutines also do not check limits, so may fail if called outside the range of the equation of state. The following two-phase routines are available.

```
      subroutine TPFL2 (t,p,z,Dl,Dv,x,y,q,ierr,herr)
      subroutine DHFL2 (D,h,z,t,p,Dl,Dv,x,y,q,ierr,herr)
      subroutine DSFL2 (D,s,z,t,p,Dl,Dv,x,y,q,ierr,herr)
      subroutine DEFL2 (D,e,z,t,p,Dl,Dv,x,y,q,ierr,herr)

c  inputs--two of the following as indicated by the first two
letters of c          the subroutine name:
c        t--temperature [K]
c        p--pressure [kPa]
c        D--bulk molar density [mol/L]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]
c
c  additional inputs--required for all routines
c        z--overall (bulk) composition [array of mol frac]
c     Dmin--lower bound on density [mol/L]
c     Dmax--upper bound on density [mol/L]
c
c  outputs:
c        t--temperature [K] (not present for TPFL2)
c        p--pressure [kPa] (not present for TPFL2)
c       Dl--molar density [mol/L] of the liquid phase
c       Dv--molar density [mol/L] of the vapor phase
c        x--composition of liquid phase [array of mol frac]
c        y--composition of vapor phase [array of mol frac]
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)
```

In the following two-phase flash routines, there is the option to pass the dew and bubble point conditions as inputs; if these data are known (from a previous call to SATT or SATP, for example), these two-phase routines will be significantly faster than the corresponding general routines described in Section C.5.1. Otherwise, the general routines will be more reliable.

```
      subroutine TDFL2
(t,D,z,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,q,ierr,herr)
      subroutine TEFL2
(t,e,z,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,q,ierr,herr)
      subroutine THFL2
(t,h,z,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,q,ierr,herr)
      subroutine TSFL2
(t,s,z,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,q,ierr,herr)
      subroutine PDFL2
(p,d,z,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
     &                  t,Dl,Dv,x,y,q,ierr,herr)
      subroutine PEFL2
(p,e,z,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
    &                 t,Dl,Dv,x,y,q,ierr,herr)
      subroutine PHFL2
(p,h,z,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
     &                  t,Dl,Dv,x,y,q,ierr,herr)
      subroutine PSFL2
(p,s,z,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
    &                 t,Dl,Dv,x,y,q,ierr,herr)
      subroutine TQFL2
(t,q,z,kq,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,ierr,herr)
      subroutine PQFL2
(p,q,z,kq,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
     &                  t,Dl,Dv,x,y,ierr,herr)

c  inputs--two of the following as indicated by the first two
letters of c          the subroutine name:
c        t--temperature [K]
c        p--pressure [kPa]
c        D--overall (bulk) molar density [mol/L]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]
c
c  additional inputs
c        z--overall (bulk) composition [array of mol frac]
c     ksat--flag for bubble and dew point limits
```

```
c           0 = dew and bubble point limits computed within
routine
c           1 = must provide values for following:
c  additional additional input--only for TQFL2 and PQFL2
c       kq--flag specifying units for input quality
c           kq = 1 quality on MOLAR basis [moles vapor/total
moles]
c           kq = 2 quality on MASS basis [mass vapor/total
mass]
c
c  additional inputs if ksat = 1
c     tbub--bubble point temperature [K] at (p,x=z)
c     tdew--dew point temperature [K] at (p,y=z)
c    --or--
c     pbub--bubble point pressure [kPa] at (t,x=z)
c     pdew--dew point pressure [kPa] at (t,y=z)
c    --and—
c    Dlbub--liquid density [mol/L] at bubble point
c    Dvdew--vapor density [mol/L] at dew point
c     ybub--vapor composition [array of mol frac] at bubble
point
c     xdew--liquid composition [array of mol frac] at dew
point
c
c  outputs--one of the following, depending on the inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c
c  additional outputs--common to all routines
c       Dl--molar density [mol/L] of the liquid phase
c       Dv--molar density [mol/L] of the vapor phase
c        x--composition of liquid phase [array of mol frac]
c        y--composition of vapor phase [array of mol frac]
c        q--vapor quality on a MOLAR basis [moles vapor/total
moles]
c           (not present for TQFL2 and PQFL2)
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)
```

```
            subroutine DQFL2
(d,q,z,kq,t,p,Dl,Dv,x,y,ierr,herr)
c
c  flash calculation given bulk density, quality, and
composition
c
c  This routine accepts only two-phase states as input.
c
c  inputs:
c        d--overall (bulk) molar density [mol/L]
c        h--overall (bulk) molar enthalpy [J/mol]
c        z--overall (bulk) composition [array of mol frac]
c       kq--flag specifying units for input quality
c           kq = 1 quality on MOLAR basis [moles vapor/total
moles]
c           kq = 2 quality on MASS basis [mass vapor/total
mass]
c
c  outputs:
c        t--temperature [K]
c        p--pressure [kPa]
c       Dl--molar density [mol/L] of the liquid phase
c       Dv--molar density [mol/L] of the vapor phase
c        x--composition of liquid phase [array of mol frac]
c        y--composition of vapor phase [array of mol frac]
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)
```

C.6 Thermodynamic Property Subroutines as f(T,ρ,x)

The following routines provide thermodynamic properties as a function of temperature, density, and composition. Typically, one or more of these will be called after finding the temperature and/or density1 with a call to one of the saturation or flash routines. Note that these routines assume that valid inputs are supplied--no range checking is performed.

```
      subroutine CRITP (x,tcrit,pcrit,Dcrit,ierr,herr)
c
c  critical parameters as a function of composition
c
c  input:
c        x--composition [array of mol frac]
c  outputs:
c    tcrit--critical temperature [K]
c    pcrit--critical pressure [kPa]
c    Dcrit--critical density [mol/L]
c     ierr--error flag:  0 = successful
c                        1 = did not converge
c     herr--error string (character*255 variable if ierr<>0)



      subroutine THERM (t,rho,x,p,e,h,s,cv,cp,w,hjt)
c
c  compute thermal quantities as a function of temperature,
density,
c  and compositions using core functions (Helmholtz free
energy, ideal
c  gas heat capacity and various derivatives and integrals)
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        p--pressure [kPa]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [J/mol-K]
c       Cv--isochoric heat capacity [J/mol-K]
c       Cp--isobaric heat capacity [J/mol-K]
c        w--speed of sound [m/s]
c      hjt--isenthalpic Joule-Thompson coefficient [K/kPa]
```

```
      subroutine THERM2 (t,rho,x,p,e,h,s,cv,cp,w,Z,hjt,A,G,
     &
xkappa,beta,dPdD,d2PdD2,dPdT,dDdT,dDdP,
     &                   spare1,spare2,spare3,spare4)
c
c  compute thermal quantities as a function of temperature,
density,
c  and compositions using core functions (Helmholtz free
energy, ideal
c  gas heat capacity and various derivatives and integrals)
c
c  this routine is similar to THERM, except that additional
properties
c  are calculated
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        p--pressure [kPa]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [J/mol-K]
c       Cv--isochoric heat capacity [J/mol-K]
c       Cp--isobaric heat capacity [J/mol-K]
c        w--speed of sound [m/s]
c        Z--compressibility factor (= PV/RT) [dimensionless]
c      hjt--isenthalpic Joule-Thompson coefficient [K/kPa]
c        A--Helmholtz energy [J/mol]
c        G--Gibbs free energy [J/mol]
c   xkappa--isothermal compressibility [1/kPa]
c           (= -1/V dV/dP = 1/rho dD/dP) [1/kPa]
c     beta--volume expansivity (= 1/V dV/dT = -1/rho dD/dT)
[1/K]
c     dPdD--derivative dP/drho [kPa-L/mol]
c   d2PdD2--derivative d^2P/drho^2 [kPa-L^2/mol^2]
c     dPdT--derivative dP/dT [kPa/K]
c     dDdT--derivative drho/dT [mol/L-K]
c     dDdP--derivative drho/dP [mol/L-kPa]
c   sparei--4 space holders for possible future properties
```

```
      subroutine ENTRO (t,rho,x,s)
c
c  compute entropy as a function of temperature, density and
composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c        s--entropy [J/mol-K]


      subroutine ENTHAL (t,rho,x,h)
c
c  compute enthalpy as a function of temperature, density, and
c  composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c        h--enthalpy [J/mol]


      subroutine CVCP (t,rho,x,cv,cp)
c
c  compute isochoric (constant volume) and isochoric (constant
pressure)
c  heat capacity as functions of temperature, density, and
composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c       cv--isochoric heat capacity [J/mol-K]
c       cp--isobaric heat capacity [J/mol-K]
```

```
      subroutine GIBBS (t,rho,x,Ar,Gr)
c
c  compute residual Helmholtz and Gibbs free energy as a
function of
c  temperature, density, and composition
c
c  N.B.  The quantity calculated is
c
c        G(T,rho) - G0(T,P*) = G(T,rho) - G0(T,rho) +
RTln(RTrho/P*)
c
c        where G0 is the ideal gas state and P* is a reference
pressure
c        which is equal to the current pressure of interest.
Since Gr
c        is used only as a difference in phase equilibria
calculations
c        where the temperature and pressure of the phases are
equal, the
c         (RT/P*) part of the log term will cancel and is
omitted.
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c       Ar--residual Helmholtz free energy [J/mol]
c       Gr--residual Gibbs free energy [J/mol]


      subroutine AG (t,rho,x,a,g)
c
c  compute Helmholtz and Gibbs energies as a function of
temperature,
c  density, and composition.
c
c  N.B.  These are not residual values (those are calculated
by GIBBS).
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        a--Helmholtz energy [J/mol]
c        g--Gibbs free energy [J/mol]
```

```
      subroutine PRESS (t,rho,x,p)
c
c  compute pressure as a function of temperature,
c  density, and composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c        p--pressure [kPa]



      subroutine DPDD (t,rho,x,dpdrho)
c
c  compute partial derivative of pressure w.r.t. density at
constant
c  temperature as a function of temperature, density, and
composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c   dpdrho--dP/drho [kPa-L/mol]



      subroutine DPDD2 (t,rho,x,dp2dD2)
c
c  compute second partial derivative of pressure w.r.t.
density at const
c  temperature as a function of temperature, density, and
composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c   d2pdD2--d^2P/drho^2 [kPa-L^2/mol^2]
```

```
      subroutine DPDT (t,rho,x,dpt)
c
c  compute partial derivative of pressure w.r.t. temperature
at constant
c  density as a function of temperature, density, and
composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c      dpt--dP/dT [kPa/K]


      subroutine DDDP (t,rho,x,drhodp)
c
c  compute partial derivative of density w.r.t. pressure at
constant
c  temperature as a function of temperature, density, and
composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c   drhodp--drho/dP [mol/L-kPa]



      subroutine DDDT (t,rho,x,drhodt)
c
c  compute partial derivative of density w.r.t. temperature at
constant
c  pressure as a function of temperature, density, and
composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c   drhodt--drho/dT [mol/L-K]
c
c   d(rho)/d(T) = -d(rho)/dP x dP/dT = -dP/dT / (dP/d(rho))
```

```
      subroutine DHDT (t,rho,x,dht)
c
c  compute partial derivative of enthalpy w.r.t. temperature
at constant
c  density as a function of temperature, density, and
composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c      dht--dH/dT [J/mol-K]


      subroutine VIRB (t,x,b)
c
c  compute second virial coefficient as a function of T & x
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        b--second virial coefficient [L/mol]


      subroutine DBDT (t,x,dbt)
c
c  compute the 2nd deriviate of B (B is the second virial
coefficient)
c  with respect to T as a function of temperature and
composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c      dbt--2nd deriviate of B with respect to T [L/mol-K]
```

```
      subroutine VIRC (t,x,c)
c
c  compute the third virial coefficient as a function of T & x
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        c--thrid virial coefficient [(L/mol)^2]
      subroutine VIRD (t,x,d)
c
c  compute the fourth virial coefficient as a function of
temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        d--fourth virial coefficient [(L/mol)^3]



      subroutine FGCTY (t,rho,x,f)
c
c  compute fugacity for each of the nc components of a mixture
by
c  numerical differentiation (using central differences) of
the
c  dimensionless residual Helmholtz energy
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c        f--array (1..nc) of fugacities [kPa]
```

```
      subroutine EXCESS (t,p,x,vE,eE,hE,sE)
c
c  compute excess properties as a function of temperature,
pressure,
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c        x--composition [array of mol frac]
c  outputs:
c        vE--excess volume [L/mol]
c        eE--excess energy [J/mol]
c        hE--excess enthalpy [J/mol]
c        sE--excess entropy [J/mol-K]
```

C.7 Transport Property Subroutine

```
      subroutine TRNPRP (t,rho,x,eta,tcx,ierr,herr)
c
c  compute the transport properties of thermal conductivity
and
c  viscosity as functions of temperature, density, and
composition
c
c  inputs:
c        t--temperature (K)
c      rho--molar density (mol/L)
c        x--composition array (mol frac)
c  outputs:
c      eta--viscosity (uPa.s)
c      tcx--thermal conductivity (W/m.K)
c     ierr--error flag:  0 = successful
c                       -31 = temperature out of range for
conductivity
c                       -32 = density out of range for
conductivity
c                       -33 = T and D out of range for
conductivity
c                       -41 = temperature out of range for
viscosity
c                       -42 = density out of range for
viscosity
c                       -43 = T and D out of range for
viscosity
c                       -51 = T out of range for both visc and
t.c.
c                       -52 = D out of range for both visc and
t.c.
c                       -53 = T and/or D out of range for visc
and t.c.
c                        39 = model not found for thermal
conductivity
c                        49 = model not found for viscosity
c                        50 = ammonia/water mix (no properties
calculated)
c                        51 = exactly at t_crit, rho_crit for a
pure
c                             fluid; tcx is infinite
c                  -58,-59 = ECS model did not converge
c     herr--error string (character*255 variable if ierr<>0)
```

C.8 Miscellaneous Properties

The following routines return the dielectric constant, melting line, and sublimation line. These properties are not available for all fluids in REFPROP.

```
      subroutine DIELEC (t,rho,x,de)
c
c  compute the dielectric constant as a function of
temperature, density,
c  and composition.
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c   output:
c       de--dielectric constant


      subroutine MELTT (t,x,p,ierr,herr)
c
c  compute the melting line pressure as a function of
temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c   output:
c        p--melting line pressure [kPa]
c        ierr--error flag:  0 = successful
c        herr--error string (character*255 variable if
ierr<>0)
```

```
      subroutine MELTP (p,x,t,ierr,herr)
c
c  compute the melting line temperature as a function of
pressure
c  and composition.
c
c  inputs:
c        p--melting line pressure [kPa]
c        x--composition [array of mol frac]
c   output:
c        t--temperature [K]
c        ierr--error flag:  0 = successful
c        herr--error string (character*255 variable if
ierr<>0)


      subroutine SUBLT (t,x,p,ierr,herr)
c
c  compute the sublimation line pressure as a function of
temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c   output:
c        p--sublimation line pressure [kPa]
c        ierr--error flag:  0 = successful
c        herr--error string (character*255 variable if
ierr<>0)


      subroutine SUBLP (p,x,t,ierr,herr)
c
c   compute the sublimation line temperature as a function of
pressure
c  and composition.
c
c  inputs:
c        p--sublimation line pressure [kPa]
c        x--composition [array of mol frac]
c   output:
c        t--temperature [K]
c        ierr--error flag:  0 = successful
c             herr--error  string  (character*255  variable  if
ierr<>0)
```

C.9 Utility Subroutines

The following "utility" routines provide fluid constants, such as the molecular weight, and provide conversions between mass and molar quantities.

```
      subroutine INFO
(icomp,wm,ttp,tnbp,tc,pc,Dc,Zc,acf,dip,Rgas)
c
c  provides fluid constants for specified component
c
c  input:
c    icomp--component number in mixture; 1 for pure fluid
c  outputs:
c       wm--molecular weight [g/mol]
c      ttp--triple point temperature [K]
c     tnbp--normal boiling point temperature [K]
c       tc--critical temperature [K]
c       pc--critical pressure [kPa]
c       Dc--critical density [mol/L]
c       Zc--compressibility at critical point
[pc/(Rgas*Tc*Dc)]
c      acf--accentric factor [-]
c      dip--dipole moment [debye]
c     Rgas--gas constant [J/mol-K]


      subroutine NAME (icomp,hname,hn80,hcas)
c
c  provides name information for specified component
c
c  input:
c    icomp--component number in mixture; 1 for pure fluid
c  outputs:
c    hname--component name [character*12]
c     hn80--component name--long form [character*80]
c     hcas--CAS (Chemical Abstracts Service) number
[character*12]


      function WMOL (x)
c
c  molecular weight for a mixture of specified composition
c
c  input:
c        x--composition array [array of mol frac]
c
c  output (as function value):
c     WMOL--molar mass [g/mol], a.k.a. "molecular weight"
      subroutine XMASS (xmol,xkg,wmix)
c
```

```
c  converts composition on a mole fraction basis to mass
fraction
c
c  input:
c     xmol--composition array [array of mol frac]
c  outputs:
c      xkg--composition array [array of mass frac]
c     wmix--molar mass of the mixture [g/mol], a.k.a.
"molecular weight"



      subroutine XMOLE (xkg,xmol,wmix)
c
c  converts composition on a mass fraction basis to mole
fraction
c
c  input:
c      xkg--composition array [array of mass frac]
c  outputs:
c     xmol--composition array [array of mol frac]
c     wmix--molar mass of the mixture [g/mol], a.k.a.
"molecular weight"


      subroutine QMOLE
(qkg,xlkg,xvkg,qmol,xl,xv,wliq,wvap,ierr,herr)
c
c  converts quality and composition on a mass basis to a molar
basis
c
c  inputs:
c      qkg--quality on mass basis [mass of vapor/total mass]
c           qkg = 0 indicates saturated liquid
c           qkg = 1 indicates saturated vapor
c           0 < qkg < 1 indicates a two-phase state
c           qkg < 0 or qkg > 1 are not allowed and will result
in warning
c     xlkg--mass composition of liquid phase [array of mass
frac]
c     xvkg--mass composition of vapor phase [array of mass
frac]
c  outputs:
c     qmol--quality on mass basis [mass of vapor/total mass]
c       xl--molar composition of liquid phase [array of mol
frac]
c       xv--molar composition of vapor phase [array of mol
frac]
c     wliq--molecular weight of liquid phase [g/mol]
c     wvap--molecular weight of vapor phase [g/mol]
c     ierr--error flag:  0 = all inputs within limits
```

```
c           -19:  input q < 0 or > 1
c     herr--error string (character*255 variable if ierr<>0)
```

```
      subroutine QMASS
(qmol,xl,xv,qkg,xlkg,xvkg,wliq,wvap,ierr,herr)
c
c  converts quality and composition on a mole basis to a mass
basis
c
c  inputs:
c     qmol--molar quality [moles vapor/total moles]
c           qmol = 0 indicates saturated liquid
c           qmol = 1 indicates saturated vapor
c           0 < qmol < 1 indicates a two-phase state
c           qmol < 0 or qmol > 1 are not allowed and will
result in warning
c       xl--composition of liquid phase [array of mol frac]
c       xv--composition of vapor phase [array of mol frac]
c  outputs:
c      qkg--quality on mass basis [mass of vapor/total mass]
c     xlkg--mass composition of liquid phase [array of mass
frac]
c     xvkg--mass composition of vapor phase [array of mass
frac]
c     wliq--molecular weight of liquid phase [g/mol]
c     wvap--molecular weight of vapor phase [g/mol]
c     ierr--error flag:  0 = all inputs within limits
c           -19:  input q < 0 or > 1
c     herr--error string (character*255 variable if ierr<>0)
```

```
      subroutine LIMITX
(htyp,t,D,p,x,tmin,tmax,Dmax,pmax,ierr,herr)
c
c  returns limits of a property model as a function of
composition
c  and/or checks input t, D, p against those limits
c
c  Pure fluid limits are read in from the .fld files; for
mixtures, a
c  simple mole fraction weighting in reduced variables is
used.
c
c  Attempting calculations below the mininum temperature
and/or above
c  the maximum density will result in an error.  These will
often
c  correspond to a physically unreasonable state; also many
equations of
c  state do not extrapolate reliably to lower T's and higher
D's.
c
c  A warning is issued if the temperature is above the maximum
but below
c  1.5 times the maximum; similarly pressures up to twice the
maximum
c  result in only a warning. Most equations of state may be
c  extrapolated to higher T's and P's.  Temperatures and/or
pressures
c  outside these extended limits will result in an error.
c
c  When temperature is unknown, set t to -1 to avoid
performing
c  the melting line check
c
c  inputs:
c     htyp--flag indicating which models are to be checked
[character*3]
c           'EOS':  equation of state for thermodynamic
properties
c           'ETA':  viscosity
c           'TCX':  thermal conductivity
c           'STN':  surface tension
c        t--temperature [K]
c        D--molar density [mol/L]
c        p--pressure [kPa]
c        x--composition array [mol frac]
c     N.B.--all inputs must be specified, if one or more are
not
c           available, (or not applicable as in case of
surface tension)
c           use reasonable values, such as:
```

```
c           t = tnbp
c           D = 0
c           p = 0
c  outputs:
c     tmin--minimum temperature for model specified by htyp
[K]
c     tmax--maximum temperature [K]
c     Dmax--maximum density [mol/L]
c     pmax--maximum pressure [kPa]
c     ierr--error flag:  0 = all inputs within limits
c                       <>0 = one or more inputs outside
limits:
c                         -1 = 1.5*tmax > t > tmax
c                          1 = t < tmin or t > 1.5*tmax
c                          2 = D > Dmax or D < 0
c                         -4 = 2*pmax > p > pmax
c                          4 = p < 0 or p > 2*pmax
c                          8 = component composition < 0 or > 1
c                              and/or composition sum < 0 or > 1
c                         16 = p>pmelt
c                        -16 = t<ttrp (important for water)
c           if multiple inputs are outside limits, ierr =
abs[sum(ierr)]
c           with the sign determined by the most severe
excursion
c           (ierr > 0 indicate an error--calculations not
possible,
c            ierr < 0 indicate a warning--results may be
questionable)
c     herr--error string (character*255 variable if ierr<>0)


      subroutine LIMITK
(htyp,icomp,t,D,p,tmin,tmax,Dmax,pmax,ierr,herr)
c
c  returns limits of a property model (read in from the .fld
files) for
c  a mixture component and/or checks input t, D, p against
those limits
c
c  This routine functions in the same manner as LIMITX except
that the
c  composition x is replaced by the component number icomp.
c  See the description of LIMITX for input and output
information.


      subroutine ERRMSG (ierr,herr)
c
c  write error messages to default output; this subroutine
should be
```

```
c  called immediately after any call to a subroutine which
potentially
c  can error out
c
c  inputs:
c     ierr--error flag:  0 = successful (no message will be
written)
c                        <0 = warning
c                        >0 = error
c     herr--error string (character*255 variable)
c
c  outputs:
c     none--error string written to default output
c
c     N.B.  the statement which writes output to the screen is
c           commented out to avoid problems with the DLL; this
must be
c           uncommented if you wish to see error messages on
the screen;
c           alternately, open a file from within your
application and
c           replace the "write (*,1000)" with "write
(unit=i,1000)" to
c           write the messages to that file (where i is a
logical unit)
```

C.10 Differences With Version 6 FORTRAN Routines

Compared to version 6, version 7 of REFPROP includes many additional routines providing additional properties and calculation options, including:

- the maximum number of components in a mixture has been increased to 20,
- the addition of dielectric constant, melting line, and sublimation line data for some fluids,
- saturation states may be specified by the density, enthalpy, or entropy in addition to the existing temperature and pressure routines,
- many additional flash combinations with virtually all combinations of temperature, pressure, density, internal energy, enthalpy, entropy, or quality are supported, and
- the flash routines with vapor quality as an input now accept quality on either a mass or molar basis.

Because of the increase in the maximum number of mixture components, the dimension of the composition arrays must be increased from 5 to 20. This is especially important for compositions which are output quantities (e.g. xliq and xvap in SATT). The composition arrays are dimensioned by the use of the PARAMETER ncmax. If you do not wish to change the size of the arrays in your application program, you may do a global change of all occurances of

PARAMETER (ncmax=20)

to PARAMETER (ncmax=5).

All version 6 subroutines are retained with the same inputs and outputs. One change to the existing routines is that the file specification variables “hfiles” and “hfmix” used in SETUP, SETKTV, and GETKTV have been changed from character*80 to character*255. This was needed to accomodate the very long path names used on some systems. These routines can detect, in most cases, if the input variable is declared as character*80, so existing programs should work without modification.

A number of common blocks holding fluid constants have been changed to accomodate new models. Users have been cautioned against refering to common blocks. The routines INFO, NAME, WMOL, etc. should be used when fluid constants are required.